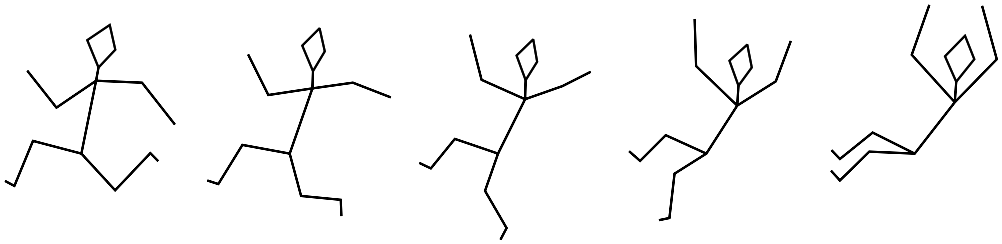
Morphing Stick Figures Using Optimized Compatible Triangulations

VITALY SURAZHSKY CRAIG GOTSMAN

Department of Computer Science
Technion—Israel Institute of Technology
Technion City, Haifa 32000, Israel

{vitus|gotsman}@cs.technion.ac.il



Abstract

A "stick figure" is a connected straight-line plane graph, sometimes called a "skeleton". Compatible stick figures are those with the same topological structure. We present a method for naturally morphing between two compatible stick figures in a manner that preserves compatibility throughout the morph. In particular, this guarantees that the intermediate shapes are also stick figures (e.g. they do not self-intersect). Our method generalizes existing algorithms for morphing compatible planar polygons using Steiner vertices, and improves the complexity of those algorithms by reducing the number of Steiner vertices used.

1 Introduction

Morphing, also known as metamorphosis, is the gradual transformation of one shape (the *source*) into another (the *target*). Morphing has wide practical use in areas such as computer graphics, animation and modeling. To achieve more spectacular, impressive and accurate results, the morphing process requires a lot of the work to be done manually. A major research challenge is to develop techniques that will automate this process as much as possible.

The morphing problem has been investigated in many contexts, e.g., morphing of two-dimensional images [4, 9, 18,19], polygons and polylines [1,5,10,14–16,18], freeform curves [13] and even voxel-based volumetric representations [6]. The morphing process always consists of solving two main problems. The first one is to find a correspondence between elements (features) of the two shapes. The second problem is to find trajectories that corresponding elements traverse during the morphing process. Regrettably, a formal definition of a successful correspondence does not exist, as well as a definition of a successful solution of the trajectory problem. In this paper we assume the more typical morphing scenario, namely, that the correspondence is given and only the trajectory problem is to be solved.

The naive approach to solve the trajectory problem is to choose the trajectories to be straight lines, where every feature of the shape travels with a constant velocity along its line towards the corresponding feature of the target during the morph. Unfortunately, this simple approach can lead to undesirable results. The intermediate shapes can vanish, i.e. degenerate into a single point, or contain self-intersections even though the source and target are simple, namely, self-intersection-free. Even if the linear morph is free of self-intersections and degeneracies, its intermediate shapes may have areas or distances between features far from those of the source and target, resulting in a "misbehaved" looking morph. Most of the research on solving the trajectory problem for morphing concentrates on trying to eliminate self-intersections and preserve the geometrical properties of the intermediate shapes. Many existing methods achieve good results for many inputs, however, almost none are able to guarantee any properties of the resulting morph.

Previous works propose methods for morphing so-called compatible planar triangulations [8, 17] and polygons [18]. Compatible means that the source and target shapes have identical topological structure in the plane, hence may be morphed in a natural manner, such that all intermediate shapes also have the same topological structure. The methods of [18] rely on the fact that a simple polygon partitions the plane into exactly two connected regions (the polygon interior and exterior). The first contribution of this paper extends those methods to morph *stick figures*, which induce much more intricate partitions of the plane. Formally, stick figures are defined as connected straight-line plane graphs. This includes shapes such as polygons, polylines, triangulations and tilings. Stick figures are used extensively by artists and animators in the entertainment industry. See the first page of the paper for an example. We show how to morph compatible stick figures by methods of morphing compatible triangulations using barycentric coordinates. This is achieved by embedding the stick figures in triangulations constructed on a set of Steiner vertices. Since it is desirable to use as few of these Steiner vertices as possible, we show how to further improve existing methods of compatible triangulation by reducing the number of Steiner vertices used. This is the second contribution of this paper.

The paper is organized as follows: Section 2 introduces the notion of compatibility and reviews previous relevant work on morphing of compatible triangulations and polygons. Section 3 shows how to morph compatible stick figures by embedding in compatible triangulations and Section 4 describes how to minimize the number of Steiner vertices used to do this. Experimental results generated by our algorithm are shown in Section 5, and we conclude in Section 6.

2 Morphing Compatible Shapes

In this section we define the notion of topological compatibility and review prior art on morphing shapes which are compatible in this sense.

2.1 Compatibility

Assume we are given a stick figure S. Since S is embedded in the plane, a clockwise cyclic ordering of the edges emanating from each vertex may be defined. This ordering may be used to decompose S into a set of *cyclic sequences*, using Algorithm 1. Two corresponding stick figures S_1 and S_2 are called *compatible* if their decompositions result in two sets of isomorphic cyclic sequences preserving the vertex correspondences. In this case we say that all sequences have the same *orientation*. Note that the geometry of the stick figures plays a role only in determining the ordering of the edges.

Once the cyclic sequences are determined, they also define a partitioning of the plane into regions. One of the sequences defines an unbounded region (having infinite area). We call this the *exterior* sequence, as opposed to the other *interior* sequences. It is possible to distinguish between the interior and exterior sequences by summing the n angles of the planar polygon on n vertices defined by the sequence. The sequence is interior iff this sum is equal to $\pi(n-2)$.

Decompose(S)
1. For each vertex v in S, order the edges incident on v in clockwise order around v.
2. Duplicate each (undirected) edge of S into two directed edges in opposite directions.
3. $i := 1$;
4. While there are unmarked edges of S:
5. Choose an arbitrary unmarked edge e of S.
6. $e_c := e$;
7. Create new empty sequence C_i;
8. While $e_c \neq e$ or $C_i =<>$ do:
9. Mark(e);
10. $C_i := C_i + t(e)$;
11. $e :=$ edge emanating from $t(e)$ and immediately following e in the clockwise order around $t(e)$.
12. end;
13. $i := i + 1$;
14. end.

Algorithm 1: Stick figure decomposition into cyclic sequences. The '$+$' operator in line 10 means to concatenate t to sequence C_i, and $t(v)$ denotes the target vertex of edge e.

When defining compatibility of two stick figures, we require that the two exterior sequences corresponding to the unbounded regions of each figure be isomorphic.

It is interesting to apply this definition to some special cases of stick figures, such as polylines, polygons and triangulations. Two polylines are compatible if the order of the corresponding vertices along the polyline are identical. The decomposition consists of one exterior sequence. Two polygons are compatible if the cyclic order of the corresponding vertices along the polygons in clockwise direction is identical. The decomposition consists of an interior sequence bounding the interior of the polygon, and an exterior sequence bounding the rest of the plane. Two triangulations consisting of t triangles are compatible iff the edges of the triangulations correspond, and the orientation of corresponding triangles is identical. See Figure 1. The decomposition consists of t interior sequences corresponding to the t triangles, and one exterior sequence corresponding to their unbounded complement.

Figure 2 shows some examples of compatible and non-

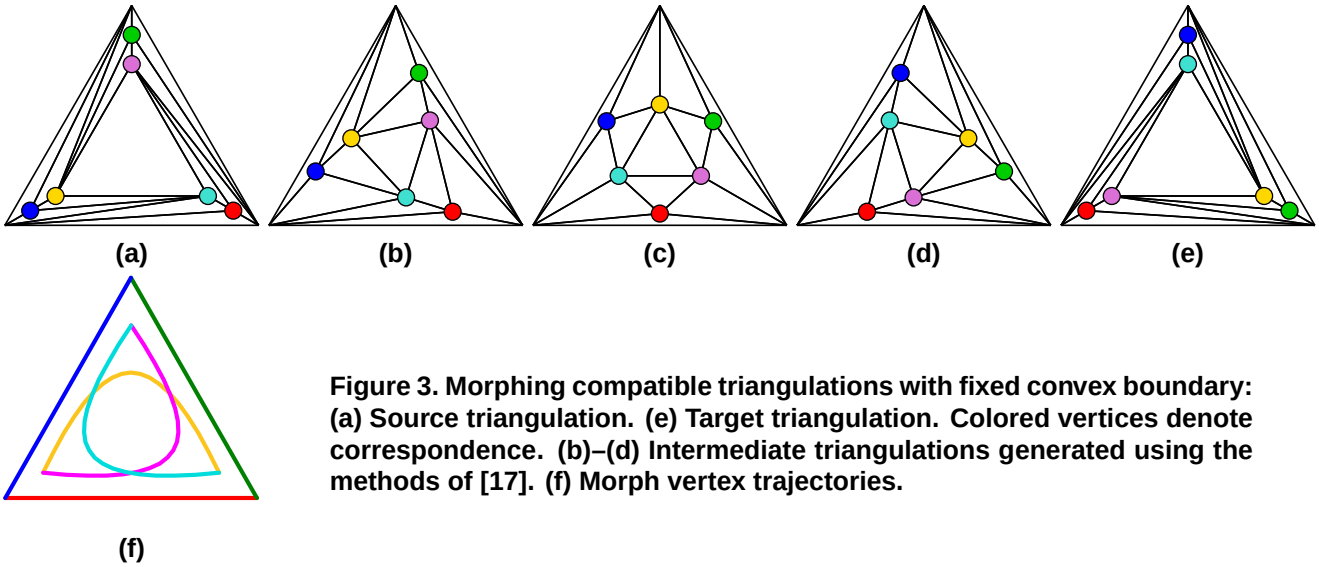


Figure 3. Morphing compatible triangulations with fixed convex boundary: (a) Source triangulation. (e) Target triangulation. Colored vertices denote correspondence. (b)–(d) Intermediate triangulations generated using the methods of [17]. (f) Morph vertex trajectories.

compatible stick figures and their decompositions into cyclic sequences.

A feature of compatible triangulations is that they define an injective piecewise-affine mapping (interpolating the point-to-point correspondence) between the convex hulls of the two point sets, without "holes" or "overlaps" in the mapping. The topic of compatible triangulations has been investigated in computational geometry, and the decision problem whether two given corresponding point sets admit a compatible triangulation is believed to be NP-Complete [12]. In this work, however, we assume they are given as compatible.

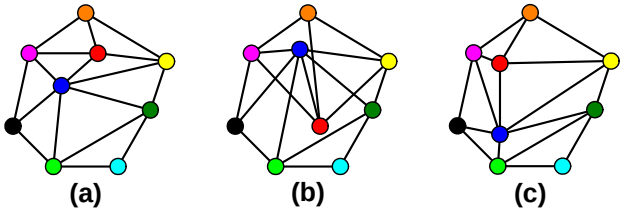


Figure 1. (a) Triangulation of 9 points in the plane. (b) Triangulation not compatible with (a). Vertex correspondence coded in colors. (c) Triangulation compatible with (a).

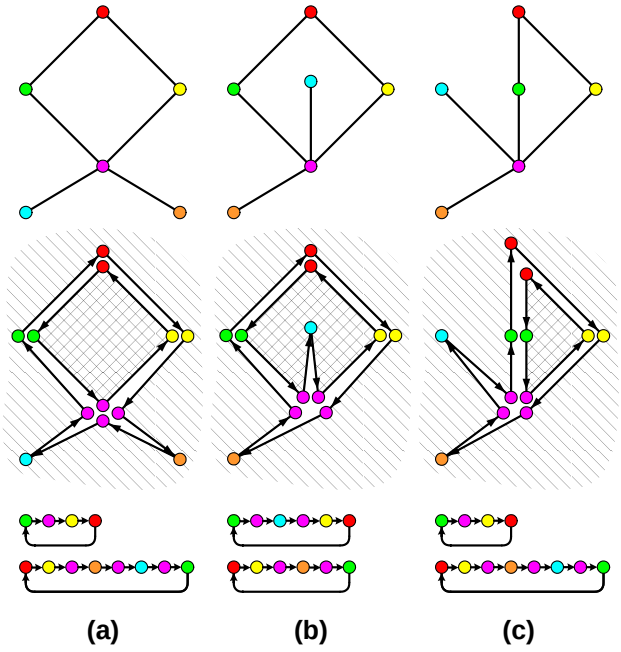


Figure 2. (a) Stick figure and decomposition to cyclic sequences. The regions of the induced partition of the plane are filled with different patterns. (b) Stick figure incompatible with (a) and its decomposition. Note that the decomposition is different from that of (a). (f) Stick figure compatible with (a) and its decomposition. Note that the decomposition is identical to that of (a).

2.2 Morphing Compatible Triangulations

Previous works have addressed the problems of morphing between compatible triangulations and polygons.

Floater and Gotsman [8] first introduced a general method for morphing between two given compatible triangulations *with a common convex boundary*. The method is based on the convex representation of triangulations using barycentric coordinates, first introduced by Tutte [20] for graph drawing purposes, and later generalized by Floater [7] for parameterization of 3D meshes. In a nutshell, this means that a legal planar triangulation on n points may be (non-uniquely) represented by a sparse $n \times n$ stochastic matrix (i.e. a non-negative matrix with unit row sums), where the i'th row contains non-zero entries only at columns associated with neighbors of the i'th vertex. The geometry (locations) of the point set is the eigenvector of the matrix with unit eigenvalue.

At the heart of the morphing method lies the principle that the morphing interpolation procedure should not be done in shape space, rather in matrix space. This avoids many of the problems associated with morphing, and basically guarantees that the triangulation remains valid (i.e. compatible with the source and target) throughout the morph. See Figure 3. The simplest morph in matrix space is linear interpolation of the corresponding matrices, since, obviously, the interpolated matrices will also be stochastic, hence represent legal triangulations. So even though the morph is quite trivial in matrix space, it can be quite complex in triangulation space, and yield very interesting results. The basic method of [8], however, is global in nature, and does not permit local control of the morph. This has been addressed recently by Surazhsky and Gotsman [17], and more local control of the individual vertex trajectories is now possible.

2.3 Morphing Compatible Polygons

Matrix space techniques always yield a valid morph and are a robust method for morphing images represented by textured triangulations. Regrettably, they are appropriate only for planar triangulations with identical convex boundaries—the most fundamental partition of the plane—hence not immediately applicable to polygon morphing.

MorphPolygons(P_1, P_2)

1. Compatibly triangulate the interiors of P_1 and P_2.
2. Enclose P_1 and P_2 by identical convex polygons Q_1 and Q_2 respectively, such that the vertices of Q_1 and Q_2 correspond.
3. Compatibly triangulate the annuli between Q_1 and P_1 and between Q_2 and P_2 respectively.
4. Morph the two compatible triangulations using the methods of [8] and/or [17].
5. Discard all edges of the triangulations not present in P_1 and P_2.

Algorithm 2: Morphing compatible polygons.

Fortunately, there is a way to reduce the problem of polygon morphing to triangulation morphing. The key is to construct compatible triangulations such that the polygons are embedded in them as a cycle of interior edges and then morph between these triangulations. Clearly, any valid morph between these two triangulations will define a valid morph of the embedded polygons. Hence, to morph two planar polygons P_1 and P_2, the general procedure is as described in Algorithm 2 (see Figure 4).

Steps 1 and 3 of Algorithm 2 are easier said than done. It is not always possible to compatibly triangulate two polygons or annuli without introducing extra vertices, also known as *Steiner* vertices. Aronov et al [2] have proposed two algorithms that construct compatible triangulations of two corresponding simple n-gons using at most $O(n^2)$ Steiner vertices. Babikov et al [3] have shown how to construct compatible triangulations of polygons with holes. However, both these methods may generate unfair triangulations as a result of producing skinny (close to degenerate) triangles. They also do not attempt to minimize the number of Steiner vertices, which is desirable, as too many Steiner vertices increase the complexity of the problem, and impose more constraints on the system.

More recently, Kranakis and Urrutia [11] and Surazhsky and Gotsman [18] proposed algorithms for compatibly triangulating two planar genus-0 or genus-1 polygons which reduce the number of Steiner vertices used. These may be used to implement Steps 1 and 3 of Algorithm 2. The method of [18] will be elaborated on in Section 4.

3 Morphing Stick Figures

Using the terminology of Section 2.1, morphing two compatible simple polygons involves compatible triangulation of the two regions corresponding to the two cyclic sequences of its decomposition—the polygon interior and the polygon exterior. The polygon exterior is circumscribed by some fixed convex enclosure to enable the triangulation in practice.

Carrying this basic idea one step further, morphing compatible stick figures is possible using the procedure described in Algorithm 3 by also considering their decompositions to cyclic sequences. Each pair of polygons defined by the interior sequences are compatibly triangulated using as few Steiner vertices as possible. The exterior sequences are circumscribed by a common convex boundary, and the annulus of these shapes (corresponding to the unbounded region) also compatibly triangulated using as few Steiner vertices as possible. The pair of unions of these triangulations are then morphed as usual. Note that the polygons corresponding to the sequences may have angles of 2π radians, but this does not affect the triangulation process. Degenerate angles (0 radians) would have been problematic,

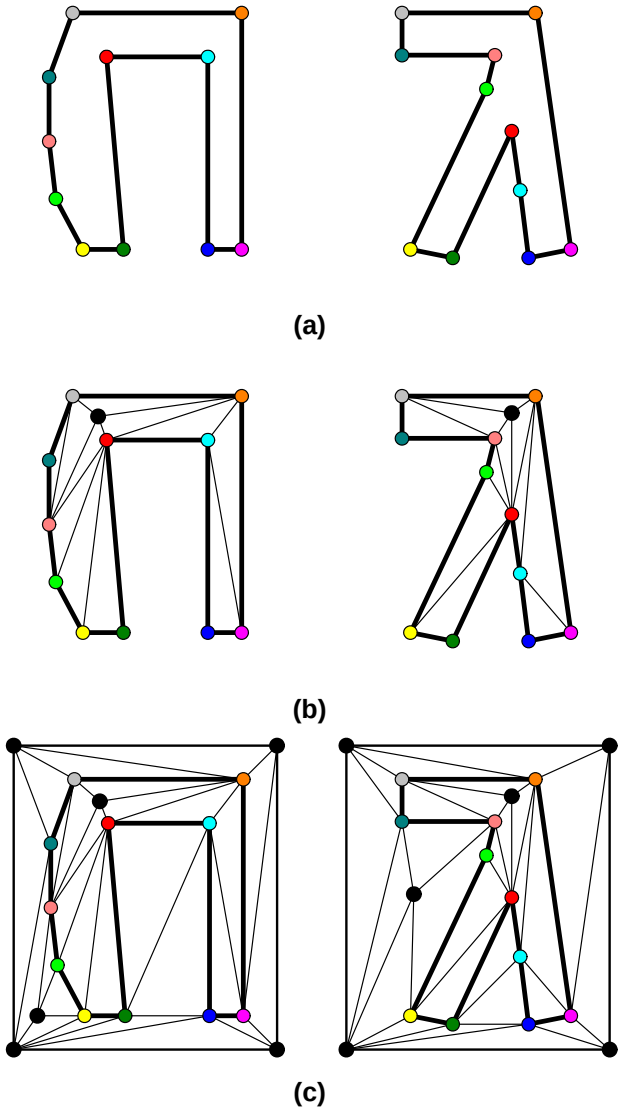


Figure 4. Embedding polygons in compatible triangulations. (a) Source and target polygons. (b) Compatible triangulations of polygons in (a). Note that one (black) Steiner vertex is used in the polygon interiors in order to achieve compatibility. (c) Each polygon is enclosed by a rectangle and the annuli compatibly triangulated. Here too one (black) Steiner vertex is used.

but cannot result from the decomposition method. A polygon containing multiple vertices with identical geometries is treated by first uniting these vertices to one vertex, triangulating the resulting polygon, and then "splitting" the vertex back to its original number.

MorphStickFigures(S_1, S_2)

1. Decompose each of S_1 and S_2 to cyclic sequences using Algorithm 1.
2. Compatibly triangulate the polygons corresponding to the interior sequences of S_1 and S_2.
3. Compatibly triangulate the region corresponding to the exterior sequence of S_1 and S_2 by circumscribing by an identical convex enclosure, cutting this annulus, and compatibly triangulating the resulting simple polygon.
4. Morph the two resulting compatible triangulations using the methods of [8] and/or [17].
5. Discard all edges of the triangulations not present in S_1 and S_2.

Algorithm 3: Morphing compatible stick figures.

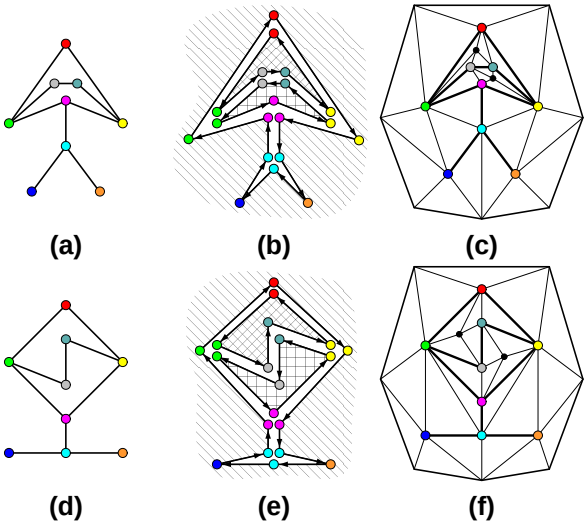


Figure 5. (a) Stick figure. (b) Decomposition of (a). (d) Stick figure compatible with (a). (e) Decomposition of (d). (c), (f) Compatible triangulations of (b) and (e) using 2 (black) Steiner vertices.

Figure 5 shows some examples of the compatible triangulations generated in this manner from a pair of stick figures.

To compatibly triangulate the two annular regions, we use a method similar to that of Babikov et al [3], where the annuli are cut, resulting in two simple polygons, which may be compatibly triangulated as usual. This contrasts with the method of [18] for compatibly triangulating two annuli in a direct manner, which was quite complicated. That method for compatible triangulation was necessary because the indirect method generated too many Steiner vertices in the vicinity of the cut. Thanks to our Steiner vertex reduction method, described in Section 4, we are able to revert to the

simple indirect method.

4 Reducing the Number of Steiner Vertices

A crucial issue in the compatible triangulation of corresponding polygons is to use as few Steiner vertices as possible. Steiner vertices are undesirable, as each extra Steiner vertex used increases the complexity of solving for the morph, and adds a compatibility constraint on the resulting morph.

In previous work, Surazhsky and Gotsman [18] showed how to reduce the number of Steiner vertices. The reduction was significant relative to the number obtained by [2], but the number of Steiner vertices generated was still very high for typical inputs. Here we present an improved version of that algorithm that reduces this ratio by an order of magnitude in most cases.

4.1 The Previous Algorithm

For completeness sake, we briefly describe here the method of [18], but refer the interested reader to that paper for details and illustrations. That method was an improvement of the second of two methods for building compatible triangulations of polygons described by Aronov et al [2], based on a so-called *universal* triangulation. This universal triangulation has the form of a *spiderweb* consisting of concentric layers of n-gons and one central vertex, where all corresponding vertices on adjacent layers are connected by radial edges, and all layers contain exactly n vertices. Any two n-gons may be compatibly triangulated using these spiderwebs, having $O(n^2)$ Steiner vertices. The variant of [18], however, may have incomplete concentric layers and, thus, different radial path lengths from the polygon vertices to the central vertex. This *reduced* spiderweb reduces the number of Steiner vertices by exploiting the shape and similarity of the two polygons.

Given a cyclic sequence of n non-negative integers, the *spiderweb connectivity* associated with this sequence is a graph of incomplete concentric cycles such that the *depth* (distance from the central vertex) of the boundary vertices is the given sequence.

The compatible triangulation algorithm of [18] proceeds as follows: Given a simple polygon, triangulate it (e.g. using the Delaunay triangulation) and consider the dual graph of the triangulation, which is always a tree. For the dual graph, locate a *central* vertex by cutting all tree leaves (equivalent to *ear-cutting* in the triangulation) simultaneously until one vertex—the center—remains. The *depth* of each triangulation face (dual vertex) is then computed as the distance from the central vertex, and subsequently *milestones* for the polygon vertices, defined to be the minimal depth associated with incident triangulation faces.

Compute a sequence of integers consisting of the maximum between the two milestones of corresponding vertices of the two polygons and use this to construct a *new* spiderweb connectivity. Then impose a triangulation on *both* input polygons using interior Steiner vertices having the same connectivity as the new common spiderweb connectivity.

The locations of the Steiner vertices are determined in a manner similar to that of [2] by "shrinking" the input polygons. This can, however, result in triangle with unattractive geometric shapes (i.e. they might be long and "skinny"). Since there is some freedom in the possible locations of these vertices, we use heuristics to achieve a better distribution of the Steiner vertices.

Note that the algorithm reduces the number of Steiner vertices relative to the universal spiderweb of [2] by choosing the depths to be the minimal allowed to accommodate both polygons (instead of $\lceil (n-5)/2 \rceil$ (when $n > 4$), as in [2]).

4.2 The Improved Algorithm

Now we show how the number of the vertices in the spiderwebs generated by the algorithm of [18] may be further reduced. Note that as that algorithm builds the reduced spiderweb for each polygon, it makes redundant vertices on layers starting from the outermost layer, namely, the input polygon. The vertices are redundant since they are positioned on straight edges of an inner polygon. As the algorithm iterates from the leaves of the dual graph towards the central vertex, the number of redundant vertices monotonically increases. At the end, it is possible to eliminate these vertices from the common reduced spiderweb if they are redundant in *both* individual spiderwebs.

The net effect of this is that the difference between the *union* and the *intersection* of the two individual spiderwebs is a spiderweb corresponding to a compatible triangulation of both polygons. This reduces the number of Steiner vertices as much as possible, while taking into account the geometric similarity of the two polygons. The Steiner vertices directly reflect the difference between the two polygon geometries. Note that in the special case where the two input polygons are identical, there will be no Steiner vertices in the compatible triangulation.

Figure 6 illustrates our improved algorithm for compatible triangulation, and Figure 7 shows compatible triangulations of two polygons using the improved method compared to the methods of [2] and [18]. The dramatic reduction in the number of Steiner vertices is quite evident.

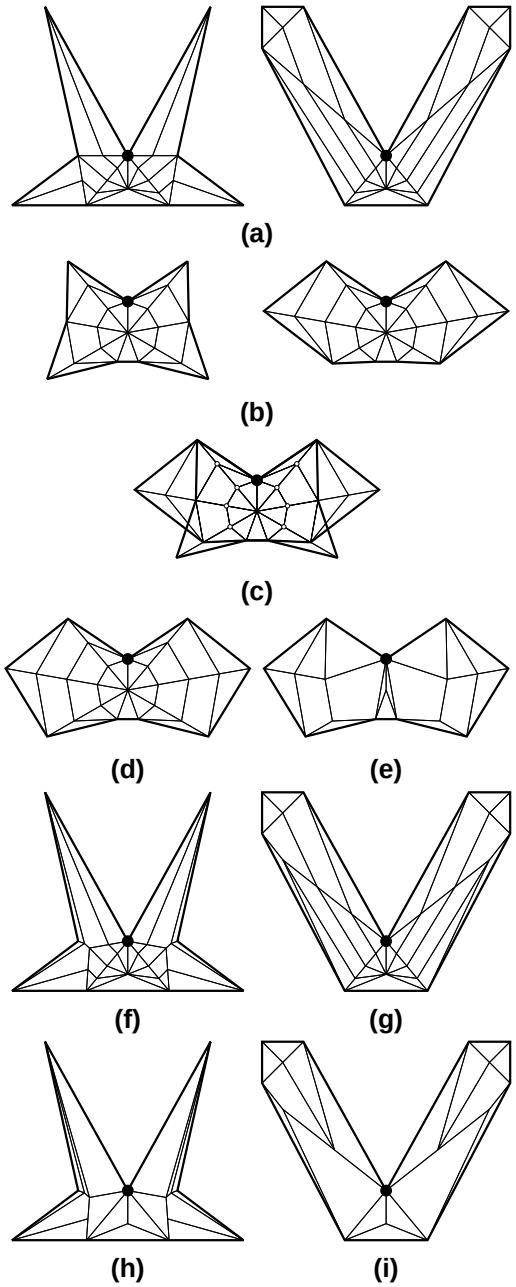


Figure 6. Construction of the improved common spiderweb. Correspondence is cyclic along the polygon starting from the black vertex. (a) Source and target polygons triangulated separately in order to define depths for vertices. (b) Spiderwebs reflecting the depths. (c) Merge of the spiderwebs of (b). The interior vertices common to both spiderwebs are marked by circles. (d) The common spiderweb of [18], which is the union of the spiderwebs of (b). (e) The improved common spiderweb is the difference between the union and the intersection. (f), (g) Compatible triangulations of the input polygons using spiderweb of (d). (h), (i) Compatible triangulations of the input polygons using spiderweb of (e).

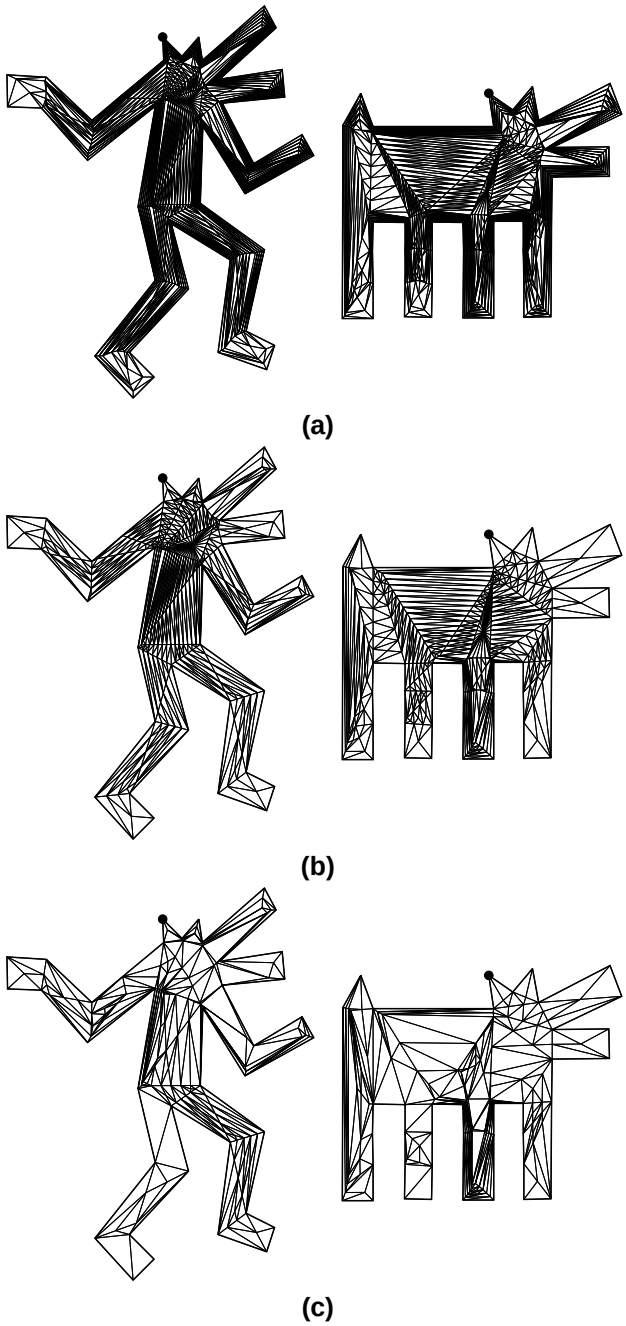


Figure 7. Compatible triangulation of two corresponding polygons of 38 vertices. Correspondence is cyclic along the polygon starting from the black vertex. (a) 575 Steiner vertices using method of [2]. (b) 334 Steiner vertices using method of [18]. (c) 153 Steiner vertices using the improved method described in Section 4.2.

5 Experimental Results

A number of sample stick figure morphs generated by our algorithm are shown in Figure 8. Each morph is illustrated by a number of snapshots during the morph. All examples used a convex hexagonal enclosure, which was a good combination of a circular symmetric shape with not too many vertices. The ratio of the number of Steiner vertices generated by the algorithm of Section 4.2 to the number of input polygon vertices did not exceed 11. The higher ratios occurred only when there was significant rotation in the morph. Animations of the examples in Figure 8 and others are available at `http://www.cs.technion.ac.il/~vitus/StickFigures`, as well as the linear morphs of the examples.

The theoretical worst case complexity of the compatible triangulation procedure described in Section 4 is $O(n^2)$, but in practice seems to be much faster. The computation is dominated by the time required to generate the intermediate frames of the morph, which involves the solution of a sparse set of linear equations. However, it did not require more than a few seconds on a 600 MHz Pentium PC to generate each of the morphs of Figure Figure 8.

6 Discussion and Conclusion

This paper has shown how to morph between two planar stick figures, which are compatible, namely have identical topologies. The generated morph is guaranteed to also be compatible, hence very natural. Special instances in the class of stick figures are planar polylines, polygons, triangulations and tilings.

The original method of morphing triangulations [8] enabled only the morphing of triangulations with an identical convex boundary. The methods reported here allow us to morph also triangulations with general (non-identical or non-convex) boundaries, as long as they are compatible—since we anyway enclose the figures in new identical convex boundaries, and triangulations are stick figures.

In real-world applications, the given source and target stick figures will rarely have the same number of vertices and edges, so it will be necessary to augment the figures with "dummy" vertices in order to apply the procedures described here. Simple methods may be employed, such as splitting edges into two, as long as the two figures are sufficiently similar in topology.

The main drawback of methods which use compatible triangulations is the number of Steiner points that they generate. In the worst case, this number can even be quadratic in the number of original vertices. A large number of vertices can be problematic, as it increases the complexity of the entire problem, and imposes many (compatibility) constraints on the system. Heuristics which will reduce as much as possible the number of Steiner points must be used. One such method was reported here, and shown to be quite effective.

There are still a number of unexplored degrees of freedom in the algorithm. For example, we use a Delaunay triangulation on the polygons when generating the compatible triangulations. A different triangulation method would lead to different results, and possibly less Steiner vertices. Moreover, a convex tiling of the polygon (vs. a full triangulation) is actually sufficient for our purposes, and using this certainly would reduce further the number of Steiner vertices. More degrees of freedom are present in the precise locations of the Steiner vertices, which have not really been optimized in the current implementation.

One of the reasons for the multitude of Steiner vertices generated for some inputs are the outer convex enclosures used, which are identical for source and target. Many of the Steiner points are required to "detect" and deal with simple shape movements, such as rotation. This could be dealt with easily without generating too many Steiner points by using a convex enclosure for the target, which is a rotated version of the convex enclosure used for the source. This would significantly improve the case where the morph contains a global rotation. However, when a few "local" rotations are at work, there is no alternative but to use methods such as ours.

Future work will address the most general case of arbitrary (disconnected) straight line plane figures.

References

[1] M. Alexa, D. Cohen-Or, and D. Levin. As-rigid-as-possible polygon morphing. *Proceedings of SIGGRAPH '2000*, pages 157–164, 2000.

[2] B. Aronov, R. Seidel, and D. L. Souvaine. On compatible triangulations of simple polygons. *Computational Geometry: Theory and Applications*, 3:27–35, 1993.

[3] M. Babikov, D. L. Souvaine, and R. Wenger. Constructing piecewise linear homeomorphisms of polygons with holes. *Proceedings of 9th Canadian Conference on Computational Geometry*, 1997.

[4] T. Beier and S. Neely. Feature-based image metamorphosis. *Computer Graphics (SIGGRAPH '92)*, 26(2):35–42, 1992.

[5] E. Carmel and D. Cohen-Or. Warp-guided object-space morphing. *The Visual Computer*, 13:465–478, 1997.

[6] D. Cohen-Or, D. Levin, and A. Solomovici. Three-dimensional distance field metamorphosis. *ACM Transactions on Graphics*, 17(2):116–141, Apr. 1998.

[7] M. S. Floater. Parameterization and smooth approximation of surface triangulation. *Computer Aided Geometric Design*, 14:231–250, 1997.

[8] M. S. Floater and C. Gotsman. How to morph tilings injectively. *Computational and Applied Mathematics*, 101:117–129, 1999.

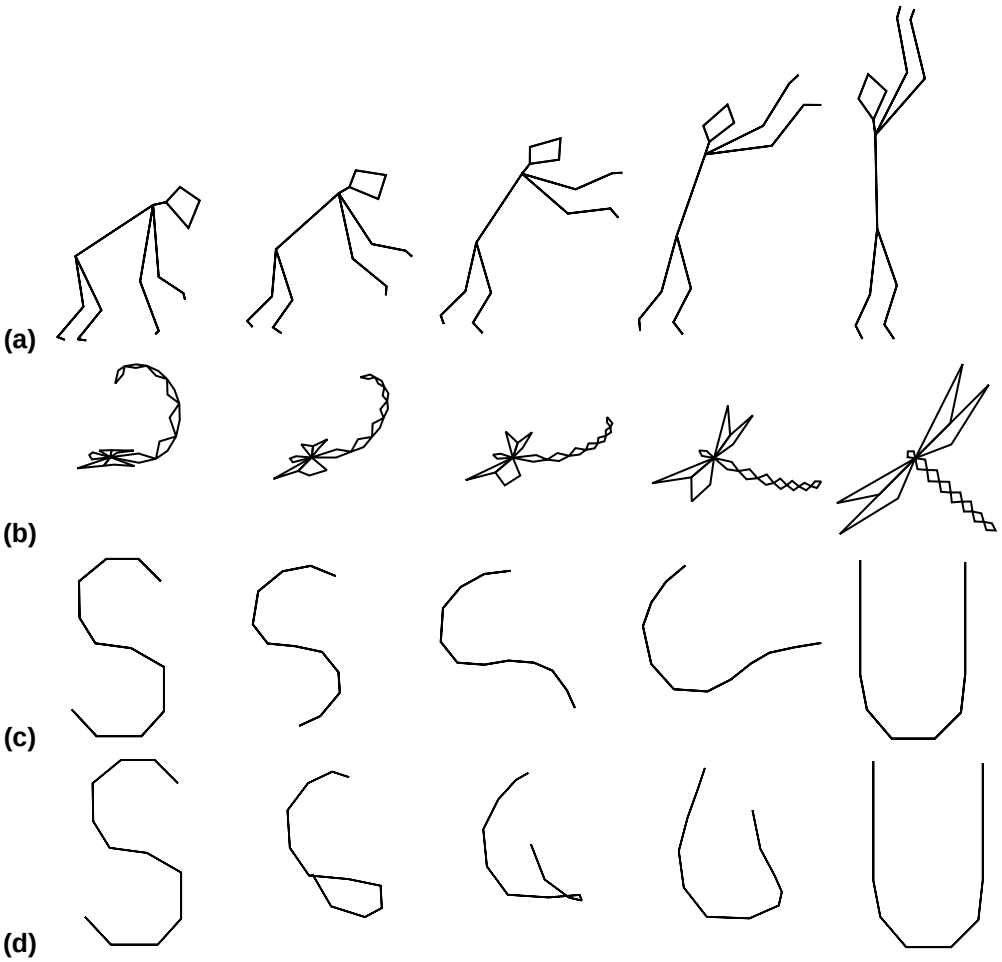


Figure 8. Morphs of some stick figures. (a) Monkey to basketball player: 18 vertices. A convex boundary of 6 vertices and 76 Steiner vertices were used. (b) Scorpion to Dragonfly. 33 vertices. A convex boundary of 6 vertices and 264 Steiner vertices were used. (c) S to U: 12 vertices. A convex boundary of 6 vertices and 127 Steiner vertices were used. (d) The linear morph between the source and the target of (c). Note the self-intersection of the intermediate stick figures generated by this straight-forward method.

[9] K. Fujimura and M. Makarov. Foldover-free image warping. *Graphical Models and Image Processing*, 60(2):100–111, Mar. 1998.

[10] E. Goldstein and C. Gotsman. Polygon morphing using a multiresolution representation. *Proceeding of Graphics Interface*, 1995.

[11] E. Kranakis and J. Urrutia. Isomorphic triangulations with small number of Steiner points. *International Jornal of Computational Geometry & Applications*, 9(2):171–180, 1999.

[12] A. Saalfeld. Joint triangulations and triangulation maps. *Proceedings of 3rd Annual ACM Symposium on Computational Geometry*, pages 195–204, 1987.

[13] T. Samoilov and G. Elber. Self-intersection elimination in metamorphosis of two-dimensional curves. *The Visual Computer*, 14:415–428, 1998.

[14] T. W. Sederberg, P. Gao, G. Wang, and H. Mu. 2D shape blending: an intrinsic solution to the vertex path problem. *Computer Graphics (SIGGRAPH '93)*, 27:15–18, 1993.

[15] T. W. Sederberg and E. Greenwood. A physically based approach to 2D shape blending. *Computer Graphics (SIGGRAPH '92)*, 26:25–34, 1992.

[16] M. Shapira and A. Rappoport. Shape blending using the star-skeleton representation. *IEEE Trans. on Computer Graphics and Application*, 15(2):44–51, 1995.

[17] V. Surazhsky and C. Gotsman. Controllable morphing of planar triangulations. To appear in ACM Transactions on Graphics, Also available at http://www.cs.technion.ac.il/∼gotsman/pubs/surazhsky.zip.

[18] V. Surazhsky and C. Gotsman. Guaranteed intersection-free polygon morphing. *Computers and Graphics*, 25(1):67–75, 2001.

[19] A. Tal and G. Elber. Image morphing with feature preserving texture. *Computer Graphics Forum (Eurographics '99 Proceedings)*, 18(3):339–348, 1999.
[20] W. T. Tutte. How to draw a graph. *Proc. London Math. Soc.*, 13:743–768, 1963.